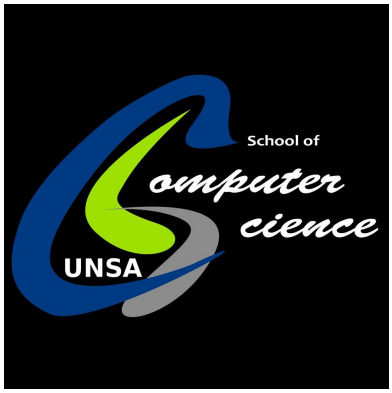

Universidad Nacional de San Agustín

Escuela Profesional de Ciencia de la Computación

Informe de QuadTree

Alumno: Villanueva Flores Luis Guillermo
Docente: Vicente Machaca Arceda

Curso: Estructuras de datos avanzada

2019

Informe teórico sobre QuadTree

25 de septiembre de 2019

1. Introducción:

En el siguiente documento se tratará la estructura de datos Quadtree, así como también se documentarán actividades propuestas.

2. ¿Qué es un Quadtree?

El término Quadtree, o árbol cuaternario, se utiliza para describir clases de estructuras de datos jerárquicas cuya propiedad común es que están basados en el principio de descomposición recursiva del espacio. En un QuadTree de puntos, el centro de una subdivisión está siempre en un punto. Al insertar un nuevo elemento, el espacio queda divido en cuatro cuadrantes. Al repetir el proceso, el cuadrante se divide de nuevo en cuatro cuadrantes, y así sucesivamente.
Una gran variedad de estructuras jerárquicas existen para representar los datos espaciales. Una técnica normalmente usada es Quadtree. El desarrollo de éstos fue motivado por la necesidad de guardar datos que se insertan con valores idénticos o similares. Este artículo trata de la representación de datos en el espacio bidimensional. Quadtree también se usa para la representación de datos en los espacios tridimensionales o con hasta 'n' dimensiones.

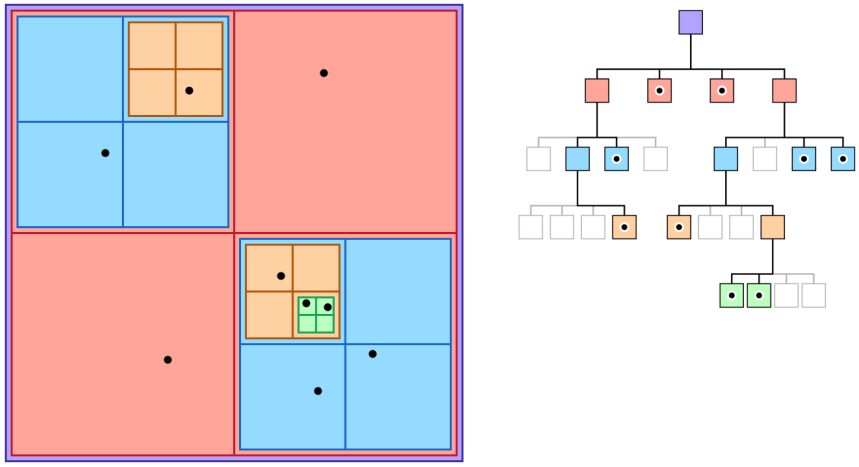

Figura 1: Representación de un Quadtree.

3. Tipos de Quadtree:

3.1. Quadtree de puntos:

El quadtree del punto es una adaptación de un árbol binario usado para representar datos de dos dimensiones del punto. Comparte las características de todos los quadtrees pero es un árbol verdadero mientras que el centro de una subdivisión está siempre en un punto. Se procesa la forma del árbol depende de los datos de la orden. Es a menudo muy eficiente en comparar los puntos de referencias pedidos de dos dimensiones, funcionando generalmente en tiempo de O (log n).[1]

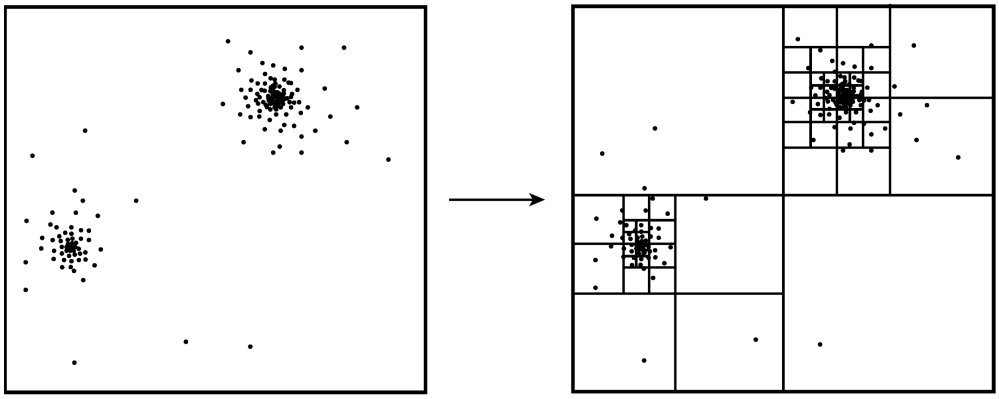

Figura 2: Representación de un Quadtree de puntos.

3.2. Quadtree de regiones:

El quadtree de la región representa una partición del espacio en dos dimensiones descomponiendo la región en cuatro cuadrantes iguales, subcuadrantes, y así sucesivamente con cada nodo de la hoja que contiene los datos que corresponden a un subregión específico. Cada nodo en el árbol tiene exactamente cuatro niños, o no tiene ningún niño (un nodo de la hoja). Un quadtree de la región con una profundidad de n se puede utilizar para representar una imagen que consiste en 2n * pixeles 2n, donde está 0 o 1 cada valor del pixel. Esta estructura de datos es unidimensional y solo se encuentra en memoria principal. Cada nodo hijo tiene asociado a él cuatro nodos, representando así los dieciséis sub-cuadrantes de dicha imagen. Una vez formado el Quadtree, los nodos hojas representan la característica de dicho píxel, que puede ser blanco o negro, si son imágenes monocromáticas, dependiendo de la uniformidad del color de los nodos hijos (si todos sus hijos son de color negro, entonces dicho nodo será representado por el color negro). Pero si algún nodo posee nodos hijos con colores no uniformes, entonces es representado por un "nodo gris". Un quadtree de la región se puede también utilizar como representación variable de la resolución de una zona de informaciones. Por ejemplo, las temperaturas en un área se pueden almacenar como quadtree, con cada nodo de la hoja almacenando la temperatura media sobre el subregión que representa.

4. Aplicaciones:

Algunas de las aplicaciones del Quadtree son:

- Manejo geográfico de diversas zonas.
- Detección de colisiones.
- Procesamiento de imagenes.

5. Actividades:

5.1. Actividad 1:

Cree un archivo main.html, este llamara a los archivos javascript que vamos a crear. El archivo p5.js es una librería para gráficos, la puede descargar de internet o se la puede pedir al profesor. En el archivo quadtree.js estará todo el código de nuestra estructura y en el archivo sketch.js estará el código donde haremos pruebas con nuestro quadtree.

Resolucion:

```
<!DOCTYPE html>
<html>
<head>
<title>QuadTree</title>
<script src="p5.min.js"></script>
<script src="quadtree.js"></script>
<script src="sketch.js"></script>
</head>
<body>
</body>
</html>
```

Listing 1: Resolución de la actividad 4(main.html)

En esta parte del código lo que se hace simplemente es llamar a nuestros demás archivos, que estos son los que se encargan de realizar las funciones para nuestro Quadtree.

5.2. Actividad 2

En el archivo quadtree.js digitemos el siguiente código, además debe completar las funciones contains y intersects (ambas funciones devuelven true o false).

Resolución:

```
class Point{
  constructor(x, y, userData){
    this.x = x;
    this.y = y;
    this.userData = userData;
  }
}

class Rectangle{
  constructor(x, y, w, h){
  this.x = x;
  this.y = y;
  this.w = w;
  this.h = h;
 }

  contains(point){
    return(point.x<=this.x+this.w && point.x>=this.x-this.w &&point.y<=this.y+this.h && point.y>=
        this.y-this.h);

  }

  intersects(range){
    return (range.x-range.w>this.x+this.w||range.x+range.w>this.x-this.w||range.y-range.h>this.y+
        this.h||range.y+range.h<this.y-this.h);

  }
}
```

Listing 2: Resolución de la actividad 2.

En esta parte defino la clase Punto, que estara conformada por un x y un y , ahí defino su constructor.Esta clase es de suma importancia debido a que esta es la fundamental para para poder insertar los mismos puntos en nuestro Quadtree.
También defino la clase rectángulo, que va a constar de 4 parámetros, un x y un y en el plano, y un w y h que significaría el ancho y largo.
Defino las funciones contains e inersects que lo que hacen basicamente es retornar si un punto esta conenido en un rectángulo y verificar si dos rectángulos se interesecan respectivamente, esto se usa al momento de insertar un punto dentro de un rectángulo.

5.3. Actividad 3:

En el archivo quadtree.js digitemos el siguiente código y complete el código de las funciones subdivide y insert. Puede revisar el pseudocódigo mostrado antes.

Resolución:

```
class QuadTree{
  constructor(boundary, n){
  this.boundary = boundary;
  this.capacity = n;
  this.points = [];
  this.divided = false;
}
subdivide(){
  var no = new Rectangle(this.boundary.x-this.boundary.w/2,this.boundary.y-this.boundary.h/2,(this
      .boundary.w)/2,(this.boundary.h)/2);
  var ne = new Rectangle(this.boundary.x+this.boundary.w/2,this.boundary.y-this.boundary.h/2,(this
      .boundary.w)/2,(this.boundary.h)/2);
  var so = new Rectangle(this.boundary.x-this.boundary.w/2,this.boundary.y+this.boundary.h/2,(this
      .boundary.w)/2,(this.boundary.h)/2);
  var se = new Rectangle(this.boundary.x+this.boundary.w/2,this.boundary.y+this.boundary.h/2,(this
      .boundary.w)/2,(this.boundary.h)/2);
  this.sonNO = new QuadTree(no, this.capacity);
  this.sonNE = new QuadTree(ne, this.capacity);
  this.sonSO = new QuadTree(so, this.capacity);
  this.sonSE = new QuadTree(se, this.capacity);
  this.divided=true;
}
insert(point){
  if(!this.boundary.contains(point))
    return;
  if(this.points.length<this.capacity){
    this.points.push(point);
  }
  else{
    if(!this.divided)
        this.subdivide();


    this.sonNO.insert(point);
    this.sonNE.insert(point);
    this.sonSO.insert(point);
    this.sonSE.insert(point);
  }
}
show(){
  stroke(255);
  strokeWeight(1);
  noFill();
  rectMode(CENTER);
  rect(this.boundary.x, this.boundary.y, this.boundary.w*2, this.boundary.h*2);
  if(this.divided){
    this.sonNO.show();
    this.sonNE.show();
    this.sonSO.show();
    this.sonSE.show();
  }
  for (let p of this.points){
```

```
    strokeWeight(4);
    point(p.x, p.y);
  }
  }
}
```

Listing 3: Resolución de la actividad 3.

En esta parte creo la clase QuadTree que es la estructura principal de nuestro programa, que tendra como varibles miembro de la clase a el perímetro, la capacidad, los puntos y un booleano que nos dirá si en caso ha sido dividido o no.
Támbien se creo la función subdivide que como su nombre lo dice sub-divide a un rectángulo grande que ya tiene al límite la cantidad de puntos en otros 4 , creando hijos al NorOeste, NorEste, SurOeste y SurEste. Para ello he calculado con una simple operación matemática en que punto y de que tamaño deberían ser los hijos, y que se muetsran en el código, por ejemplo para el hijo NorOeste he restado el punto x de nuestro rectángulo principal menos la mitad del ancho , para obtener un punto, el otro se calculo restando el punto y menos la mitad del alto, y así ya tenemos los dos puntos, ahora para saber sus ancho y alto respectivamente lo que hice fue simplemente al ancho actual dividirlo entre dos y lo mismo con el alto.
Así como ha sido para el hijo NorOeste , se hace para los demás pero considerando el hijo que queramos crear.
Támbien se hizo la función insert, que verifica si un punto puede ser insertado en un rectángulo, osea si lo contiene , si en caso puede ser se inserta y si no se verifica si ha sido subdividido para poder insertarlo, o sino subdividirlo e insertarlo. Y por último la función show que se encarga de mostrar los puntos insertados y la subdivisiones en nuestra estructura.

5.4. Actividad 4:

Editemos el archivo sketch.js. En este archivo estamos creando un QuadTree de tamaño 400x400 y en él estamos insertando 3 puntos. Ejecute la aplicación.
Resolución:

```
let qt;
let count=0;
function setup(){
  createCanvas(400,400);
  let boundary = new Rectangle(200,200,200,200); //centr point and half of width and height
  qt = new QuadTree(boundary, 4); //each section just could have 4 elements
  console.log(qt);
  for(let i=0;i<3;i++){
        let p=new Point(Math.random()*400,Math.random()*400);

  }
  background(0);
  qt.show();
}
```

Listing 4: Resolución de la actividad 3.

En esta parte de código lo que hace básicamente es crear un cuadrado con una dimensión definida y a partir de ella se inserta los puntos en nuestro QuadTree, para este caso se creo 3 puntos aleatoriamente y se insertó en nuestro QuadTree, que dicho sea de paso fue creado con las dimensiones de nuestro cuadrado que definimos más arriba. La Representación del resultado se puede observar a continuación.

Figura 3: Ejecución de la aplicación.

5.5. Actividad 5:

Abra las opciones de desarrollador (opciones/más herramientas/ opciones de desarrollador) de su navegador para visualizar la console. Muestre sus resultados.
Resolución:

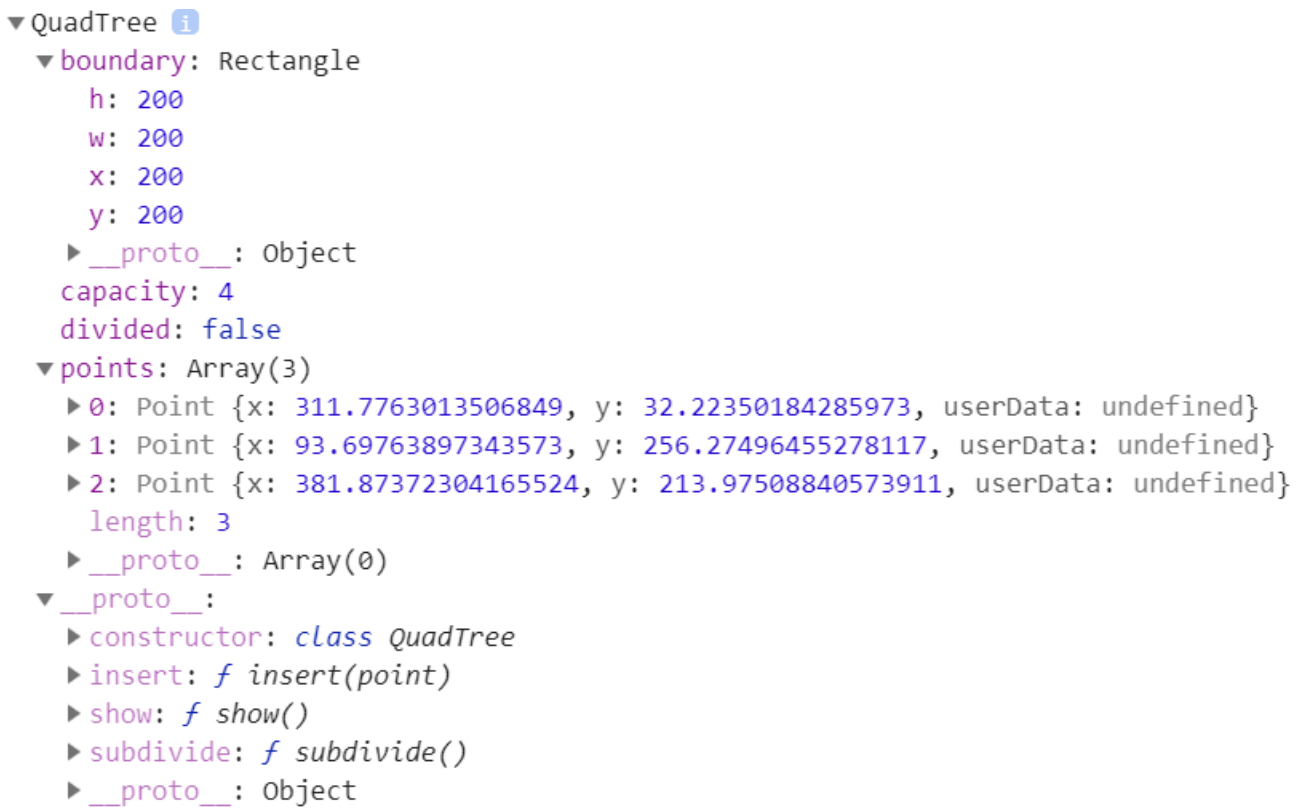

Figura 4: Resultados de la ejecución.

Aquí se puede observar como es que nuestro ordenador nos muestra las dimensiones que creamos para nuestro cuadrado y también los puntos en que parte fueron insertados, esto básicamente es para darnos cuenta como es que nuestro ordenador ha tomado nuestras instrucciones.

5.6. Actividad 6:

Inserte más puntos y muestre cómo varían sus resultados.
Reoslución:

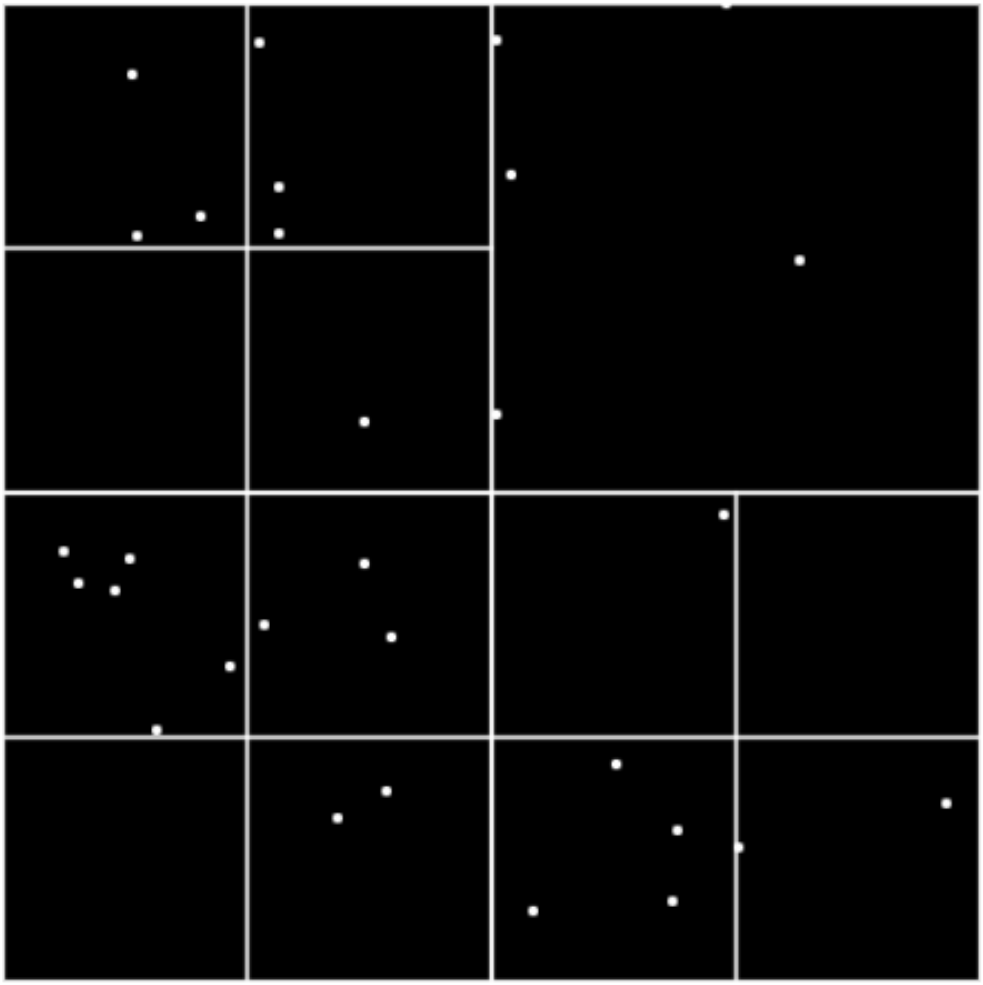

Figura 5: Resultados de la ejecución.

Vemos como al momento de insertar más puntos nuestra estructura automáticamente se subdivide en más cuadrados y repite este proceso siempre y cuando el número de puntos no sea mayor a 4 en cada cuadrado que ha creado con la subdivisión.

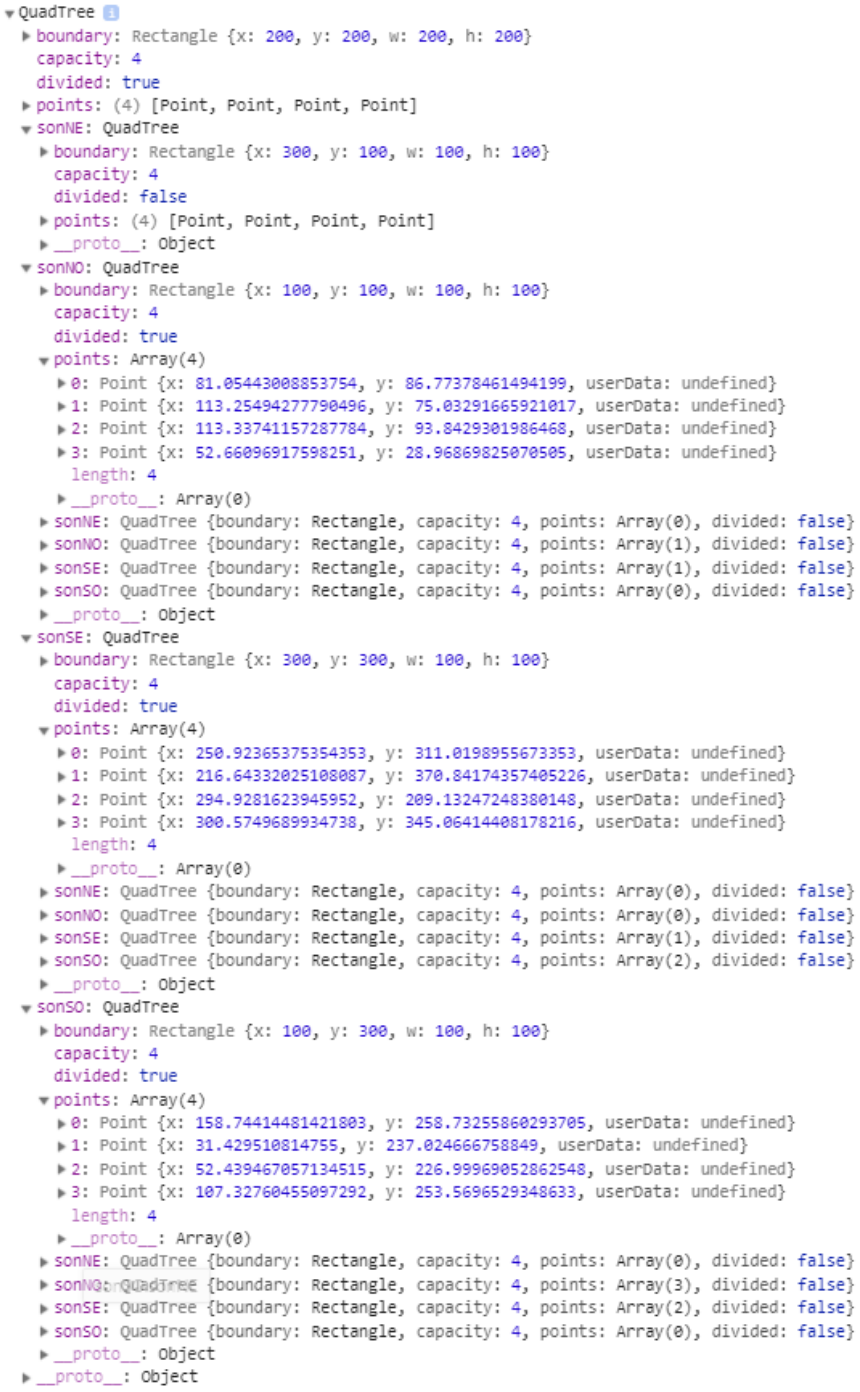

Figura 6: Ejecución de la aplicación.

Igual que en la parte de arriba,se muestra una representación de la consola para verificar las dimensiones de los nuevos cuadrados creados,así como en que coordenadas han sido insertados nuestros puntos.

5.7. Actividad 7:

Edite el archivo sketch.js con el siguiente código. En este caso nos dala posibilidad de insertar los puntos con el mouse. Muestre sus resultados y comente cómo funciona el código.
Resolución:

```
function sleep(n){

  var now =new Date().getTime();
  while(new Date().getTime()<now+n)
  {

  }
}

function setup(){
  createCanvas(400,400);
  let boundary = new Rectangle(200,200,200,200); //centr point and half of width and height
  qt = new QuadTree(boundary, 4); //each section just could have 4 elements
}
function draw(){
  background(0);
    if (mouseIsPressed){
       sleep(200);
       for (let i = 0; i < 1; i++){
        let m = new Point(mouseX + random(-5,5), mouseY + random(-5,5));
        qt.insert(m)
     }
  }
  background(0);
  qt.show();
}
```

Listing 5: Código para colocar puntos con el mouse.

En esta parte he creado la función sleep que lo hace es que no grafique puntos tan rápido al momento de hacer un click, sino que se espere un tiempo, ya que sin esta función el programa al hacer un click graficaba rapidamente unos 5 puntos y para evitar ello se creo esta función.También esta la función Setup que crea nuestro QuadTree y la función Draw que es la que se encarga de al momento de hacer un click cree un punto, pero ahora ya no los cree tan rapido si no que cree un punto a la vez.

5.8. Conclusiones

La primera conclusión que llegué es que esta estructura nos servirá mucho cuando se trabaje con gráficos, ya que si vemos a la imagen por cuadrantes entonces podemos realizar muchas aplicaciones con ella, no solo considerando puntos sino también regiones como de color, y a partir de ella darle funcionalidades.

También me di cuenta que esta estructura se podría representar como una matriz, y donde haya un punto significa que ahí hay un dato osea en nuestra matriz.

Básicamente esto lo veo enfocado a la computación gráfica pero también veo que puede servir al momento de querer guardar pocos datos , así puedo usar algunas regiones para almacenarlo y no gastar memoria insuficiente.

Si lo veo a nivel espacial, nos ayudaría para saber em que coordenada de un determinado espacio se encuentra un punto cualquiera, para hacer esto el QuadTree sería muy importante, para no perder referencia en que parte se coloco el punto o algún dato en específico.